基于 EM 算法的混合高斯模型参数估计实验报告

19231178 于天贺

一、实验目标

数据使用链接中的代码身高数据，这里假设每一个子分布的身高数据服从高斯分布，使用 EM 算法来估计高斯混合模型的参数，并使用这些参数来进行预测。同时对模型进行评估，并解释模型的性能。

二、实验原理

2.1.混合高斯模型

混合高斯模型是一种概率模型，用于描述由多个高斯分布组成的数据集。在混合高斯模型中，假设数据集中的每个样本都是从多个不同高斯分布中随机生成的，每个高斯分布称为一个混合成分，因此混合高斯模型又称为高斯混合模型（Gaussian Mixture Model，GMM）。

混合高斯模型的基本思想是，将数据集中的每个样本视为一个随机变量，该随机变量服从多个高斯分布中的一个，其具体服从哪个高斯分布是由一个隐变量（即混合成分指示变量）控制的。该隐变量的取值是一个离散随机变量，每个可能的取值对应着一个混合成分，因此混合高斯模型实际上是一个带有隐变量的概率生成模型。

混合高斯模型通常用来对复杂的数据分布进行建模和估计，如图像、音频、文本等数据。利用 EM 算法可以对混合高斯模型进行参数估计，从而实现数据的聚类和分类等任务。

2.2.EM 算法

EM 算法（Expectation-Maximization Algorithm）是一种迭代的优化算法，用于在含有隐变量的概率模型中求解最大似然估计或最大后验概率估计。EM 算法的基本思想是，先通过观测数据估计出隐变量的后验概率分布，然后利用这个后验概率分布对模型参数进行更新，不断迭代直到收敛为止。

EM 算法的核心是两步迭代：E 步（Expectation Step）和 M 步（Maximization Step）。在 E 步中，先利用当前的模型参数估计隐变量的后验概率分布；在 M 步中，利用这个后验概率分布重新估计模型参数。这两个步骤交替进行，直到收敛为止。

EM 算法通常用于求解含有隐变量的概率模型的参数估计，如高斯混合模型、隐马尔可夫模型等。EM 算法具有收敛性和渐进正确性等优点，在实际应用中得到了广泛的应用。但是，由于 EM 算法是一种局部优化算法，其结果取决于初始值的选择，因此需要根据具体情况进行调参和选择初始值。

2.3.使用 EM 算法估计高斯混合模型参数

（1）初始化参数。设高斯混合成分的个数为 K，则需要初始化每个高斯混合成分的混合系数π_k，均值μ_k 和协方差矩阵Σ_k。

（2）E 步（Expectation Step）。计算隐变量的后验概率$p(z_k = 1|x)$，即给定观测值x，第k个混合成分生成该观测值的概率。根据贝叶斯公式，可以得到：

$$p(z_k = 1|x) = \frac{\pi_k N(x|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x|\mu_j, \Sigma_j)}$$

其中$N(x|\mu_k, \Sigma_k)$表示多元高斯分布的概率密度函数：

$$N(x|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} exp\left\{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right\}$$

（3）M 步（Maximization Step）。估计高斯混合模型的参数，即混合系数 π_k 和每个高斯分布的均值 μ_k 和协方差矩阵 Σ_k。对于混合系数，可以使用观测样本和隐变量的后验概率来估计：

$$\pi_k = \frac{1}{N} \sum_{i=1}^{N} p(z_k = 1|x_i)$$

对于每个高斯分布，可以使用观测样本和隐变量的后验概率来估计均值和协方差矩阵：

$$\mu_k = \frac{\sum_{i=1}^{N} \pi_k x_i}{\sum_{i=1}^{N} \pi_k}$$

$$\Sigma_k = \frac{\sum_{i=1}^{N} \pi_k (x_i - \mu_k)(x_i - \mu_k)^T}{\sum_{i=1}^{N} \pi_k}$$

（4）重复执行 E 步和 M 步，直到收敛。可以定义一个收敛准则，例如固定迭代次数、两次迭代参数之间的距离小于某个阈值等。

整个 EM 算法的流程可以简单地描述为：从初始化的参数开始，反复执行 E 步和 M 步，直到参数收敛为止。在 E 步中，根据当前的参数值计算每个观测样本属于每个混合成分的后验概率。在 M 步中，根据这些后验概率重新估计混合系数、均值和协方差矩阵的值。每次迭代后都会计算对数似然函数值，并将其与上一次迭代的值进行比较，如果两者之差小于一个给定的阈值，则认为参数已经收敛，否则继续迭代。

三、实验过程

首先进行数据的导入。由于数据是代码生成的，并同时写入 excel 文件，所以这里为了简化步骤，不再从 excel 导入数据，直接将生成数据的代码拿来即用。代码如下：

```
# 定义高斯分布的参数
mean1, std1 = 164, 3
mean2, std2 = 176, 5
# 从两个高斯分布中生成各 50 个样本
data1 = np.random.normal(mean1, std1, 500)
data2 = np.random.normal(mean2, std2, 1500)
data = np.concatenate((data1, data2), axis=0)
```

之后进行混合高斯模型参数的初始化，每个高斯混合成分的混合系数，初始都设置为 $1/k$。对于每个高斯混合成分的均值，在所有混合数据样本中随机选择一个样本数据作为均值。对于每个高斯混合成分的方差，全部初始化为 1。具体代码如下：

```
# 初始化参数
K = 2   # 混合高斯分布的个数
```

```
pi = np.ones(K) / K    # 混合系数
mu = np.random.choice(data, size=K)    # 均值
sigma = np.ones(K)    # 方差
```

之后使用 EM 算法进行迭代更新混合高斯模型的参数，需要注意，迭代次数的设定，需要一定的技巧，这里我通过画图的方式，先后选取了几个迭代次数进行尝试，当混合高斯模型的参数基本不在发生变化时，才能证明模型已经收敛，才能说明迭代次数参数的设定才是合理的。这里通过尝试不同迭代次数并观察模型参数变化，最终确定迭代次数为 300。具体代码如下：

```
pi_lst1 = [pi[0]]
mu_lst1 = [mu[0]]
sigma_lst1 = [sigma[0]]
pi_lst2 = [pi[1]]
mu_lst2 = [mu[1]]
sigma_lst2 = [sigma[1]]

# EM 算法迭代更新参数
for i in range(300):
    # E 步：计算后验概率
    post_prob = np.zeros((K, len(data)))
    for k in range(K):
        post_prob[k] = pi[k] * norm.pdf(data, loc=mu[k], scale=sigma[k])
    post_prob /= post_prob.sum(axis=0)
    # M 步：更新参数
    for k in range(K):
        pi[k] = post_prob[k].mean()
        mu[k] = (post_prob[k] * data).sum() / post_prob[k].sum()
        sigma[k] = np.sqrt((post_prob[k] * (data - mu[k]) ** 2).sum() /
post_prob[k].sum())
```

```
    pi_lst1.append(pi[0])
    mu_lst1.append(mu[0])
    sigma_lst1.append(sigma[0])
    pi_lst2.append(pi[1])
    mu_lst2.append(mu[1])
    sigma_lst2.append(sigma[1])
```

得到训练好的模型的参数后，需要进行数据准备画图，为了画混合高斯模型函数以及高斯子模型函数，需要在所有数据的最大最小值之间选择 num_point 个点作为 x，同时根据混合高斯模型以及高斯子模型，分别求出每个函数的函数值。同时为了方便模型进行预测，需要求出两个子模型之间的分类边界 boundary。具体代码如下：

```
num_point = 100
boundary = 0
x = np.linspace(start=data.min(), stop=data.max(), num=num_point)
y_total = np.zeros_like(x)

for k in range(K):
    y_total += pi[k] * norm.pdf(x, loc=mu[k], scale=sigma[k])
y_single_lst = [pi[k] * norm.pdf(x, loc=mu[k], scale=sigma[k]) for k in range(K)]

flag = y_single_lst[0][0] > y_single_lst[1][0]
for i in range(1, num_point):
    if flag != (y_single_lst[0][i] > y_single_lst[1][i]):
        boundary = x[i]
        break
```

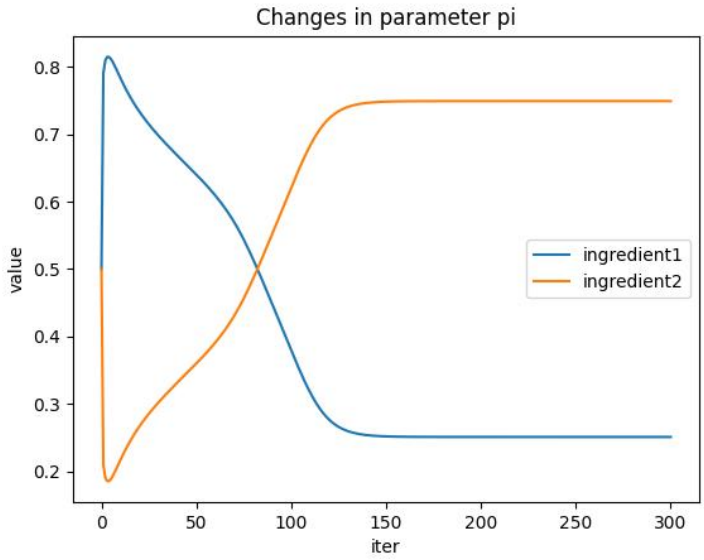
在数据准备完成之后进行画图。首先画出模型参数（每个高斯混合成分的混合系数、均值和方差）随迭代次数的变化图。之后画出原始身高数据的直方图与混合高斯模型曲线图的对比图，观察训练好的混合高斯模型是否在一定程度上拟合原始数据。之后分别画出混合高斯模型的每个成分的子模型函数，以及边界线。具体代码如下：

```
# 绘制结果
plt.figure(1)
plt.plot([i for i in range(0, len(pi_lst1))], pi_lst1, label='ingredient1')
plt.plot([i for i in range(0, len(pi_lst2))], pi_lst2, label='ingredient2')
plt.xlabel('iter')
plt.ylabel('value')
plt.title('Changes in parameter pi')
plt.legend()
plt.show()

plt.figure(2)
plt.plot([i for i in range(0, len(mu_lst1))], mu_lst1, label='ingredient1')
plt.plot([i for i in range(0, len(mu_lst2))], mu_lst2, label='ingredient2')
plt.xlabel('iter')
plt.ylabel('value')
plt.title('Changes in parameter mu')
plt.legend()
plt.show()

plt.figure(3)
plt.plot([i for i in range(0, len(sigma_lst1))], sigma_lst1, label='ingredient1')
plt.plot([i for i in range(0, len(sigma_lst2))], sigma_lst2, label='ingredient2')
plt.xlabel('iter')
plt.ylabel('value')
```

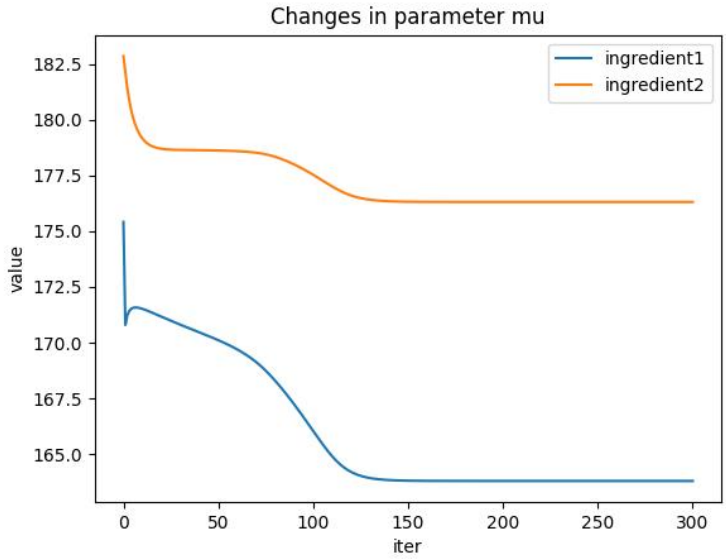
```
plt.title('Changes in parameter sigma')
plt.legend()
plt.show()


plt.figure(4)
plt.hist(data, bins=20, alpha=0.5, density=True, label='data')
plt.plot(x, y_total, 'r-', linewidth=2, label='mixture gaussian')
plt.legend()
plt.xlabel('Height (cm)')
plt.ylabel('Probability')
plt.title('Gaussian Mixture Model')
plt.show()


plt.figure(5)
for k in range(K):
        plt.plot(x, y_single_lst[k], label='ingredient'+str(k+1))
plt.axvline(x=boundary, color='red', label='boundary: y='+str(boundary))
plt.legend()
plt.xlabel('Height (cm)')
plt.ylabel('Probability')
plt.show()
```

四、实验结果与分析

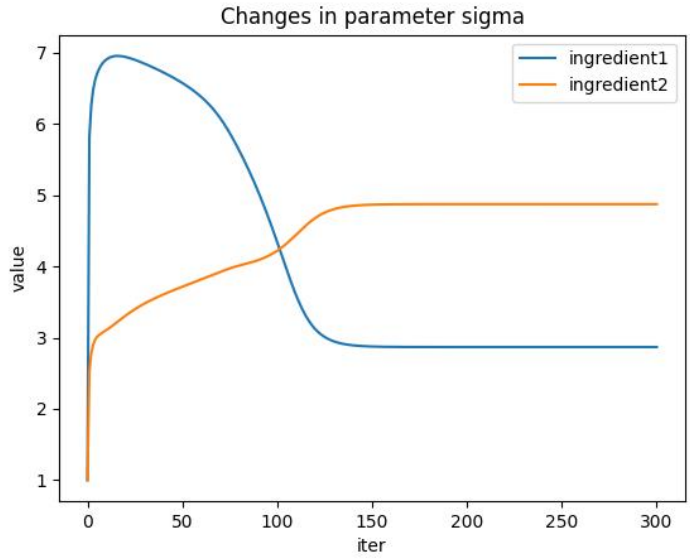
对于两个高斯混合成分的混合系数变化，如图所示，当选择总的迭代次数为 300 时，在大概 130 次迭代时达到稳定状态，分别为 0.75 与 0.25。符合原始数据两个分布数据量 1500：500 的比例。



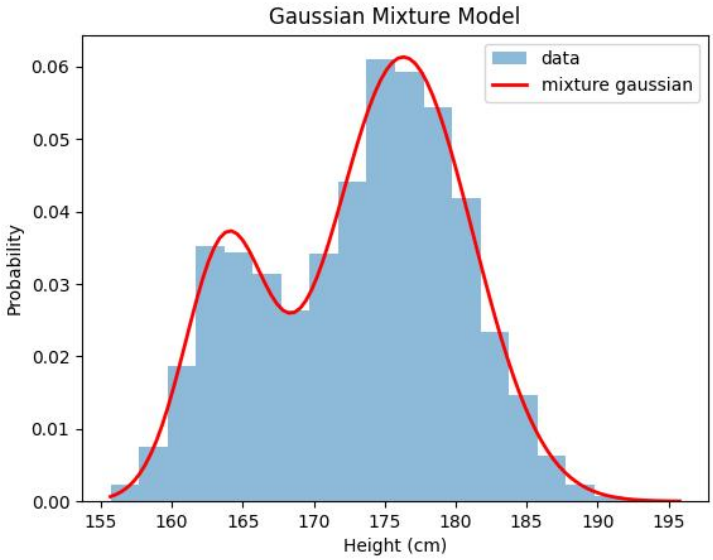
对于两个高斯混合成分的均值变化，如图所示，当选择总的迭代次数为 300 时，在大概 130 次迭代时达到稳定状态，大致分别为 176 与 164，与原始数据的两个均值基本一致。



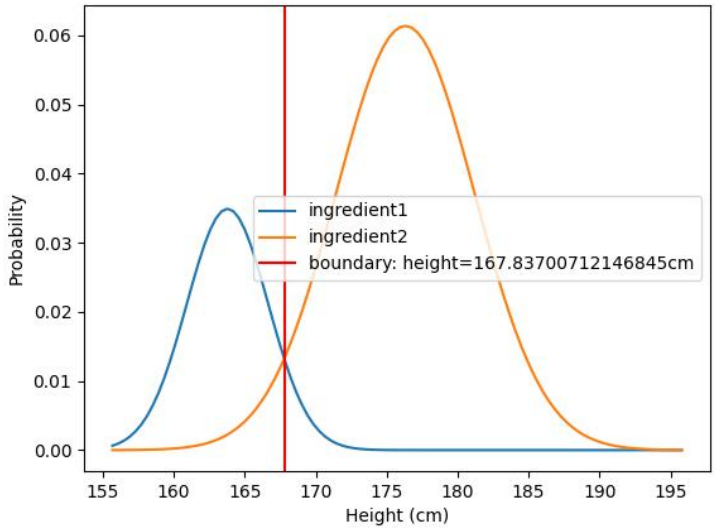
对于两个高斯混合成分的方差变化，如图所示，当选择总的迭代次数为 300 时，在大概 130 次迭代时达到稳定状态，大致分别为 5 与 3，与原始数据的两个方差基本一致。



将原始数据的分布直方图（纵坐标归一化后）以及得到的混合高斯模型函数同时在一个图上画出，如图所示，可见高斯混合模型基本拟合原始数据，经过 EM 算法训练得到的混合高斯模型比较准确。



将每个高斯混合成分的函数图在一个图片上画出，如图所示，可以求出当 x=167.837 时，样本分别属于两个成分的概率大小比较会发生变化，这个分界线即是用于预测分类的阈值。



五、模型的预测

再次按照训练集数据的生成方式生成测试集，即生成两组数据，第一组数据 500 个，服从均值为 164，方差为 3 的正态分布，第二组数据 1500 个，服从均值为 176，方差为 5 的正态分布。

预测过程即将所有数据与训练得到的子模型分界线 167.837 进行比较，小于阈值即预测为第一组，大于则预测为第二组。

具体预测代码如下：

```
np.random.seed(2)
data3 = np.random.normal(mean1, std1, 500)
np.random.seed(3)
data4 = np.random.normal(mean2, std2, 1500)
print((np.count_nonzero(data3 < boundary) +
       np.count_nonzero(data4 >= boundary))/2000)
```

最终得到预测准确率为 93.35%。